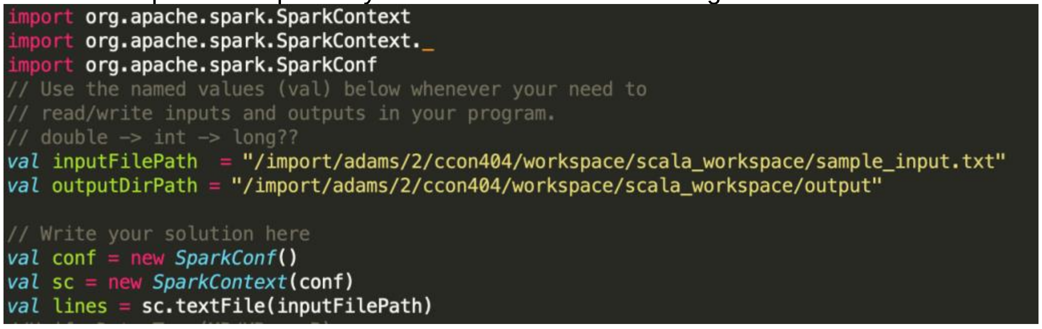
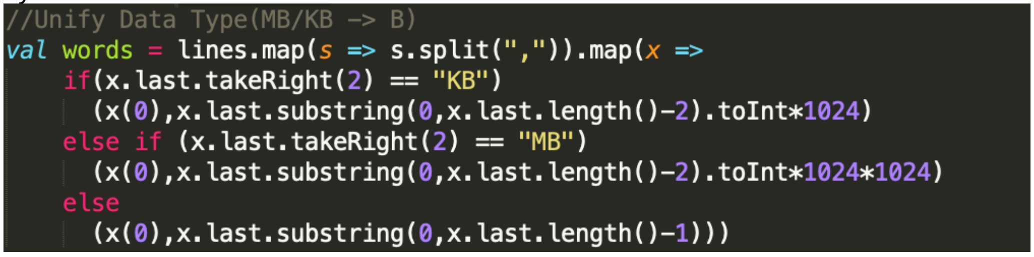
This short report will explain my solution to COMP9313 assignment2



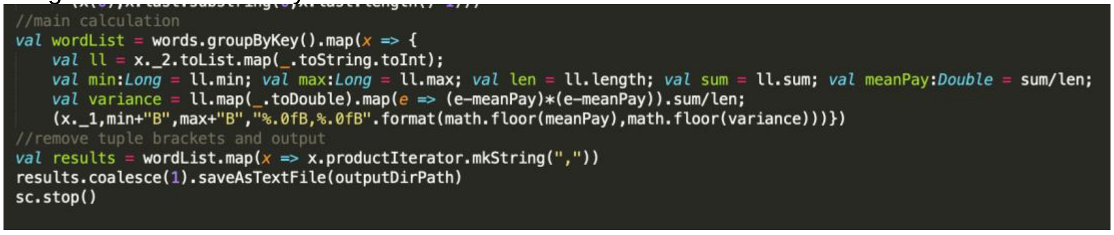
```scala
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
import org.apache.spark.SparkConf
// Use the named values (val) below whenever your need to
// read/write inputs and outputs in your program.
// double -> int -> long??
val inputFilePath  = "/import/adams/2/ccon404/workspace/scala_workspace/sample_input.txt"
val outputDirPath = "/import/adams/2/ccon404/workspace/scala_workspace/output"

// Write your solution here
val conf = new SparkConf()
val sc = new SparkContext(conf)
val lines = sc.textFile(inputFilePath)
```

In the first part of my solution, there are some initial set ups and some variables such as inputFilePath and outpputDirPath are defined for further use. Moreover, variable input files are read and captured by variable lines.



```scala
//Unify Data Type(MB/KB -> B)
val words = lines.map(s => s.split(",")).map(x =>
    if(x.last.takeRight(2) == "KB")
      (x(0),x.last.substring(0,x.last.length()-2).toInt*1024)
    else if (x.last.takeRight(2) == "MB")
      (x(0),x.last.substring(0,x.last.length()-2).toInt*1024*1024)
    else
      (x(0),x.last.substring(0,x.last.length()-1)))
```

In the second part, I try to convert all different payloads from unit of MB and KB to B. I did this by checking the last two characters of the last element of the input line, if these two character are “MB”, then the rest characters of the last element of the input file will be converted to integer and timed by 1024*1024, however, if the last two characters are “KB”, the rest characters will be converted to Integer and then timed by 1024.



```scala
//main calculation
val wordList = words.groupByKey().map(x => {
    val ll = x._2.toList.map(_.toString.toInt);
    val min:Long = ll.min; val max:Long = ll.max; val len = ll.length; val sum = ll.sum; val meanPay:Double = sum/len;
    val variance = ll.map(_.toDouble).map(e => (e-meanPay)*(e-meanPay)).sum/len;
    (x._1,min+"B",max+"B","%.0fB,%.0fB".format(math.floor(meanPay),math.floor(variance)))})
//remove tuple brackets and output
val results = wordList.map(x => x.productIterator.mkString(","))
results.coalesce(1).saveAsTextFile(outputDirPath)
sc.stop()
```

The last part of my solution is mainly about calculating the desired value max, min, mean and variance. After the groupByKey() function is called, the outputs are in the form of (**key, CompactBuffer(payload)).** Firstly, I convert CompactBuffer(payload) to a list of values of integer types. Secondly, I am able to compute the max, min, mean and variance values of the payloads of corresponding http address. Finally, formatting the results and output it into a tuple.

The variable wordlist is contained by a groups of tuples, if outputting them directly the outputDirPath, the results will be wrong, as they contains brackets as well. In order to remove the bracket and keep the content only, I write another map function which uses the **productIterator()** function of each tuple object to obtain the tuple elements and concatenate them together to obtain the desired results.